**4ème année AE – SE INSA**

**Stage technicien au LAAS :  
Simulink Parrot AR Drone Support improvements**

|  |
| --- |
| **Guide d’utilisation du projet** *SimulinkARDroneTarget-master* |

**par :   
ROY Raphaël**

# Table des matières

[Table des matières 2](#_Toc434829776)

[I. Mise en place de l’environnement de travail 4](#_Toc434829777)

[1) Compilateur 4](#_Toc434829778)

[2) MATLAB ® 4](#_Toc434829779)

[3) Environnement de développement GNU linux (connexion au drone) 4](#_Toc434829780)

[Solution 1 : Oracle VM VirtualBox – Ubuntu 4](#_Toc434829781)

[Solution 2 : Installer Cygwin (MinGW n’a pas la bonne version de make) 4](#_Toc434829782)

[II. Prise en main progressive 5](#_Toc434829783)

[1) Méthodologie 5](#_Toc434829784)

[2) Compilation avec Sourcery CodeBench 5](#_Toc434829785)

[3) Connexion au drone 5](#_Toc434829786)

[4) Chargement et exécution d’un programme sur le drone 6](#_Toc434829787)

[5) Génération simple de code avec MATLAB ® 6](#_Toc434829788)

[6) Génération, chargement et exécution de code, avec MATLAB ® 6](#_Toc434829789)

[III. Composition du projet 7](#_Toc434829790)

[1) \AR\_Drone\_Models\Hello\_World : simple test 7](#_Toc434829791)

[2) \AR\_Drone\_Models\Flight\_Models : contrôle/commande du drone 7](#_Toc434829792)

[AR\_Drone\_Flight\_Control 7](#_Toc434829793)

[Modes de fonctionnement possibles 7](#_Toc434829794)

[Control Panel 7](#_Toc434829795)

[Data Panel 8](#_Toc434829796)

[Flight Panel 8](#_Toc434829797)

[Actuators Panel 8](#_Toc434829798)

[AR\_Drone\_Mission\_Height 8](#_Toc434829799)

[AR\_Drone\_Host\_Receive 8](#_Toc434829800)

[3) \AR\_Drone\_Models\Calibration : calibration de la balise inertielle 8](#_Toc434829801)

[Calibration rapide de l’accéléromètre et du gyromètre 8](#_Toc434829802)

[Calibration à 360 degrés de l’accéléromètre 9](#_Toc434829803)

[Calibration du gyromètre (+ température) 9](#_Toc434829804)

[Calibration du magnétomètre 10](#_Toc434829805)

[4) \AR\_Drone\_Target : génération de code sur la cible 10](#_Toc434829806)

[IV. Possibilités d’amélioration 10](#_Toc434829807)

# Mise en place de l’environnement de travail

## Compilateur

→ Sourcery CodeBench for ARM GNU/Linux

1. Télécharger une version datant d’avant l’été 2014 (version Lite gratuite et illimitée dans le temps) ici : <http://sourcery.mentor.com/public/gnu_toolchain/arm-none-linux-gnueabi/>
2. Installer dans un dossier proche de la racine du disque dur de l’OS, et vérifier qu’il n’y a pas d’espace dans le chemin d’accès (ex : C:\Sourcery\_CodeBench convient très bien)
3. Lors de l’installation, choisir « Modify PATH for current user »

## MATLAB ®

→ MATLAB ® R2014a

1. Extraire le contenu de l’archive dans un répertoire proche de la racine d’un disque dur, sans qu’il n’y ait d’espace dans le chemin d’accès.
2. Lancer le script « install\_script.m » contenu dans le répertoire « AR\_Drone\_Target » extrait.
3. Sélectionner le répertoire bin contenu dans le répertoire d’installation de Sourcery CodeBench (ex : C:\Sourcery\_CodeBench\bin)

## Environnement de développement GNU linux (connexion au drone)

### Solution 1 : Oracle VM VirtualBox – Ubuntu

1. Dans l’onglet « Configuration » de la machine virtuelle, modifier les paramètres « Réseau » :  
   Changer le mode d’accès réseau en « Accès par pont », et sélectionner la case « Câble branché »
2. Pour être relié à la connexion Wifi de Windows (pour se connecter au drone par exemple), sélectionner sa carte Wifi dans la liste déroulante « Nom » (juste en dessous de « Mode d’accès réseau »
3. Pour être relié au câble ethernet (pour une connexion ftp filaire par exemple), sélectionner sa carte réseau filaire utilisée…
4. Vérifier que l’on est connecté à l’appareil avec lequel on veut échanger (par Wifi ou ethernet)
5. Vérifier qu’on est bien connecté au réseau au sein de la machine virtuelle, sinon cliquer sur l’icône de réseau en haut à droite, puis sur « Auto eth0 ».

### Solution 2 : Installer Cygwin (MinGW n’a pas la bonne version de make)

1. Télécharger un setup de Cygwin et l’exécuter en tant qu’administrateur
2. Choisir d’installer à partir d’Internet
3. L’installer à la racine du disque dur de l’OS (par exemple, C:/Cygwin)
4. Indiquer de stocker les fichiers d’installation dans le dossier précédent (par exemple, C:\Cygwin\Installation\_files)
5. Choisir un serveur au hasard
6. Lors de la sélection des packages, développer la catégorie « development tools » (« Devel ») en cliquant sur « + », et sélectionner les packages suivant, en cliquant sur « Skip » :
   * gcc-core: C compiler
   * gcc-g++: C++ compiler
   * gdb: The GNU Debugger
   * make: the GNU version of the 'make' utility
7. Développer la catégorie « Net » et sélectionner le package « inetutils » (pour obtenir l’outil telnet) en cliquant sur « Skip », puis cliquer sur « Next »
8. Rajouter le répertoire C:\Cygwin\bin à la variable d’environnement Path (système) :
   * Clic-droit sur l’icône du Poste de travail → « Propriétés » → onglet « Avancé »
   * Clic sur « Variables d’environnement » → sélectionner « Path »
   * Clic sur « Modifier » et rajouter « ;C:\Cygwin\bin »
9. Copier l’exécutable « make.exe » du dossier « C:\Cygwin\bin » sur le bureau
10. Le renommer « gmake.exe » et le coller dans le répertoire de MATLAB où se situe le gmake (par exemple, C:\Program Files\MATLAB\R2014a\bin\win64), en remplaçant l’ancien.

# Prise en main progressive

## Méthodologie

1. Compiler du code simple avec Sourcery CodeBench
2. Se connecter au drone et accéder à ses données
3. Importer du code simple compilé sur le drone et l’exécuter
4. Générer du code simple à partir de MATLAB (compilé)
5. Générer du code (compilé) de MATLAB + le charger sur le drone + l’exécuter
6. Essayer avec le programme principal de contrôle du drone

## Compilation avec Sourcery CodeBench

1. Réaliser un fichier source (en C ou en C++) de type « Hello World » (utiliser par exemple le projet d’allumage des LEDS qui est censé affecter 1 possibilité à chaque LED)
2. Utiliser le makefile fourni (produit à .elf à partir de plusieurs fichiers .cpp)
3. Modifier le makefile si besoin, afin de préciser le bon chemin du compilateur : par exemple « C:\Sourcery\_CodeBench\bin\ »

## Connexion au drone

1. Se connecter au réseau WiFi du drone (pour tester la connexion : ping 192.168.1.1)
2. Pour accéder aux données, écrire dans un terminal (sous linux) ou dans un invité de commande (sous Windows, si CygWin a été installé) :

telnet 192.168.1.1 (192.168.1.1 étant l’@IP du drone)

→ Si aucun environnement linux n’est présent, on peut aussi écrire dans l’invité de commande sous Windows :

“C:\Program Files\MATLAB\R2014a\toolbox\idelink\foundation\hostapps\plink.exe” –telnet –P 23 192.168.1.1

## Chargement et exécution d’un programme sur le drone

1. Ouvrir le terminal unix (ou l’invité de commande sous Windows si on a CygWin…)
2. Charger le fichier en écrivant …

Sous linux, dans un terminal :

* + cd <chemin\_du\_répertoire\_du\_fichier>
  + ftp
  + open 192.168.1.1 (met les fichiers dans le répertoire data/video du drone)

OU open 192.168.1.1 5551 (met les fichiers dans le répertoire upload du drone)

* + put <nom\_du\_fichier>.elf
  + disconnect
  + quit (ou <exit> si la commande n’est pas reconnue)

1. Exécuter le programme en écrivant :
   * telnet 192.168.1.1 (environnement linux installé)

OU <repertoire\_de\_plink>\plink.exe –telnet –P 23 192.168.1.1 (sous Windows)

* + cd data/video (OU cd update si on a fait un ftp avec le port 5551)
  + chmod +x <nom\_du\_fichier>.elf
  + ./<nom\_du\_fichier>.elf

## Génération simple de code avec MATLAB ®

1. Ouvrir un fichier Simulink correspondant à une application
2. Dans l’onglet « Simulation », cliquer sur « Model Configuration Parameters »
3. Vérifier que dans l’onglet « Solver », l’option « Higher priority indicates higher task priority » soit bien cochée
4. Pour générer du code et compiler, sans pour autant charger et exécuter le programme sur le drone, aller sur l’onglet « Coder Target » de la fenêtre « Model Configuration Parameters » : cliquer sur « Build options » et choisir « Build » au lieu de « Build, load and run »
5. Cliquer sur le bouton « Build »
6. Si une erreur du type « gcc: No input files // gmake: \*\*\* [nom\_du\_fichier\_à\_créer] Error 1» apparaît, c’est que le make intégré à MATLAB est de la version 3.81. Pour remédier à cela, télécharger un make d’une version supérieure, le copier sur le bureau, le renommer gmake.exe, et le mettre à la place du gmake.exe présent dans le répertoire de la version de MATLAB utilisée (exemple, pour R2014a : C:\Program Files\MATLAB\R2014a\bin\win64)  
   → Si Cygwin est installé, récupérer le make.exe et cela suffira (tant que Cygwin est installé)  
   → Si MinGW est installé, récupérer le mingw32-make.exe et les fichiers libgcc\_s\_dw2-1.dll, libiconv-2.dll et libintl-8.dll du répertoire C:\MinGW\bin (le make a besoin des 3 fichiers)

→ pour plus d’infos : <https://bugzilla.mozilla.org/show_bug.cgi?id=345482>

## Génération, chargement et exécution de code, avec MATLAB ®

1. Vérifier que l’option « Build, load and run » est sélectionnée (fenêtre « Model Configuration Parameters » : liste déroulante « Build options » de l’onglet « Coder Target »)
2. Vérifier que dans l’onglet « Solver », l’option « Higher priority indicates higher task priority » soit bien cochée
3. Vérifier qu’on est bien connecté au drone par WiFi
4. Cliquer sur « Build »

# Composition du projet

## \AR\_Drone\_Models\Hello\_World : simple test

Un fichier Simulink simple censé tourner sur le drone et afficher un compteur jusqu’à 10s sur un invité commande de la machine compilant le fichier Simulink. A compiler en mode « Normal ».

## \AR\_Drone\_Models\Flight\_Models : contrôle/commande du drone

Ce répertoire contient le fichiers Simulink principal de contrôle du drone, ainsi que les fichiers qui lui sont directement liés.

### AR\_Drone\_Flight\_Control

→ C’est le fichier Simulink de contrôle du drone. C’est la source-même de l’exécutable qui sera sur le drone pour réaliser le projet.

#### Modes de fonctionnement possibles

1. Mode « External » : le programme s’exécute sur l’ordinateur (communication pour les données des capteurs et pour les commandes des actionneurs)
   * Utiliser la référence « AR\_Drone\_ConfigSet » (paramètres du modèle)
   * Activer l’interrupteur UDP/External pour être en « External »
   * Compiler (+ charger & exécuter)
   * Cliquer sur l’icône de connexion hôte-cible
   * Cliquer sur le bouton de Simulation (flèche verte)
   * => Commander le drone par ce même Simulink, via le « Command Panel »
2. Mode « Normal » : le programme s’exécute sur le drone (communication UDP optionnelle)
   * Utiliser la référence « AR\_Drone\_ConfigSet\_NOEXT » (paramètres du modèle)
   * Activer l’interrupteur UDP/External pour être en « UDP »
   * Compiler (+ charger & exécuter)
   * Si voulu, il est possible de mettre en place une communication UDP pour récupérer des données (pour information), pour atterrir d’urgence ou pour relancer la mission : ouvrir le fichier Simulink « AR\_Drone\_Host\_Receive » et lancer sa simulation (flèche verte) => la commande s’effectue via le « Command Panel » Control Panel

#### Command Panel

#### Data Panel

#### Flight Panel

#### Actuators Panel

### AR\_Drone\_Mission\_Height

Il s’agit d’un fichier Simulink pouvant être rattaché au Simulink principal, par référence. Il est censé indiquer à ce fichier quelle mission de vol exécuter. « AR\_Drone\_Mission\_Height » a pour but de maintenir le drone à la hauteur spécifiée par l’utilisateur.

### AR\_Drone\_Host\_Receive

C’est le fichier à exécuter lorsqu’on veut passer en mode « UDP », c’est-à-dire exécuter le programme en totalité sur le drone, et lui envoyer des consignes simples par communication UDP. Il permet également d’observer les données du système.

## \AR\_Drone\_Models\Calibration : calibration de la balise inertielle

→ Ce répertoire contient des fichiers permettant de mettre à jour les constantes de calibration dont le fichier Simulink principal (AR\_Drone\_Flight\_Control) se sert pour rectifier les mesures.

→ Dans un premier temps, utiliser le fichier principal (AR\_Drone\_Flight\_Control) avec comme consigne « Auto-Calibration\_ON » dans le « Control Panel ». Si le drone semble juste dériver et qu’il ne se comporte pas complètement aléatoirement, seule la calibration rapide (« Auto\_quick\_calib ») devrait suffire. Il sera par la suite possible de faire décoller le drone sur n’importe quel sol, en indiquant la consigne « Auto-Calibration\_OFF ».

### Calibration rapide de l’accéléromètre et du gyromètre

→ dossier « \AR\_Drone\_Models\Calibration\Auto\_quick\_calib »

1. Lancer le script « AR\_Drone\_Startup\_Script »
2. Regarder si le sol est bien horizontal en faisant décoller le drone (en utilisant le fichier Simulink de contrôle de vol classique, et en précisant Auto-Calibration\_ON)
3. Laisser le drone sur le sol le plus horizontal possible
4. Lancer le Simulink « Get\_Horizontal\_Measures » (Build + Connect to the target + Run)  
   (si demandé, changer les paramètres de configuration : cocher « Higher priority value indicates higher task priority »)
5. Attendre que les scopes affichent des valeurs toutes les 3 secondes. Arrêter la simulation quand souhaité (après s’être assuré que les nouvelles valeurs des scopes étaient proches des anciennes), et relever ces valeurs
6. Vérifier que les valeurs sont bonnes en relançant le fichier Simulink de contrôle de vol classique, et en précisant Auto-Calibration\_OFF (essayer sur un sol moins horizontal)

### Calibration à 360 degrés de l’accéléromètre

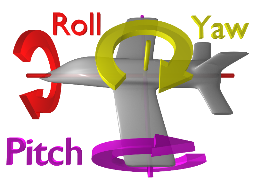
→ dossier « \AR\_Drone\_Models\Calibration\Accelerometer\_calib »

→ ne fonctionne que sur MATLAB® R2015a pour l’instant …

1. Lancer le script « AR\_Drone\_Startup\_Script »
2. Répéter (6 fois) les étapes suivantes :
   * Mettre le drone dans une position particulière
   * Lancer le Simulink « Record\_accelerometer » (Build + Connect to target + Run)
   * Attendre que la simulation se termine correctement, c’est-à-dire que les lignes suivantes doivent apparaître sur la console Windows :
     + \*\*terminating model\*\*
     + #
     + # 3
   * Enregistrer les données : save(‘Measures/Recorded\_Accel\_Z\_up’,’Recorded\_Accelerometer’)
   * Lancer le Simulink « PlayBack\_Data » (simulation)
   * Mettre à jour le script « Calculate\_Offset\_Values »
3. Lancer le script « Calculate\_Offset\_Values »
4. Mettre à jour et lancer le script « Save\_Calibration\_Parameters.m » (dossier parent)
5. Vérifier la calibration en relançant le Simulink« Record\_accelerometer » (Build + Connect to target + Run) pour une position particulière

### Calibration du gyromètre (+ température)

→ dossier « \AR\_Drone\_Models\Calibration\GyrometerTemp\_Offset\_Calib »

1. Lancer le script « AR\_Drone\_Startup\_Script »
2. Mettre le drone à plat et le laisser stable
3. Lancer le Simulink « TempMeasure » (Build + Connect to the target + Run)  
   (si demandé, changer les paramètres de configuration : cocher « Higher priority value indicates higher task priority »)
4. Attendre que la simulation se termine correctement, c’est-à-dire que les lignes suivantes doivent apparaître sur la console Windows :
   * \*\*terminating model\*\*

* #
* # 3

1. Lancer le Simulink « TempCorrect\_Offline » (simulation)
2. Lancer le script « Temperature\_Testing \_Script »
3. Mettre à jour et lancer le script « Save\_Calibration\_Parameters.m » (dossier parent)
4. Vérifier la calibration en relançant le Simulink « TempCorrect\_Offline »

### Calibration du magnétomètre

→ dossier « \AR\_Drone\_Models\Calibration\Magnetometer\_calib »

→ ne fonctionne que sur MATLAB® R2015a pour l’instant …

1. Lancer le script « AR\_Drone\_Startup\_Script »
2. Mettre le drone à plat et le laisser stable
3. Lancer le Simulink « MeasureMag » (Build + Connect to the target + Run)
4. Attendre que la simulation se termine correctement, c’est-à-dire que les lignes suivantes doivent apparaître sur la console Windows :
   * \*\*terminating model\*\*

* #
* # 3

1. Lancer la commande Matlab « plot\_and\_calibrate\_mag(MagData) »
2. Mettre à jour et lancer le script « Save\_Calibration\_Parameters.m » (dossier parent) :
   * La ligne « Ellipsoid center » correspond aux Offsets
   * La ligne « Ellipsoid radii » correspond aux Gains

## \AR\_Drone\_Target : génération de code sur la cible

1. +codertarget : fichiers de configuration de la cible
2. Blocks : fichiers sources du projet (+ en-têtes)
3. Include : fichiers en-tête concernant l’ordonnanceur
4. Registry : paramètres de compilation → pour modifier un paramètre du makefile :
   * Faire les modifications dans le fichier « gcc\_codesourcery\_arm\_linux\_gnueabihf.m »
   * Exécuter le fichier, puis exécuter le fichier « save\_new\_mat.m »
5. Src : fichiers sources

# Possibilités d’amélioration

1. Asservissement du Yaw
2. Récupérer les angles absolus, la vitesse (linéaire) du drone, et sa position relative
3. Asservissement en vitesse, puis en position => amélioration du contrôle (par PIDs)
4. Créer des bus d’entrée et de sortie pour le Simulink de mission
5. Récupérer les flux vidéos (en couleur) : projet TicTacToe ?
6. Capteurs : → Caractériser les données obtenues (qualité du signal, dispersion, bruit)  
    → Intégrer les mesures provenant du magnétomètre et du baromètre
7. Essayer la compilation avec la toolchain de Linaro (Linaro GCC) qui est gratuite (support package cortex a)